Algebraic Approach to Promise Constraint Satisfaction*

Jakub Bulín
Department of Algebra, Fac. Math. & Phys., Charles University
Prague, Czechia
jakub.bulin@mff.cuni.cz

Andrei Krokhin
Department of Computer Science, Durham University
Durham, UK
andrei.krokhin@durham.ac.uk

Jakub Opršal
Department of Computer Science, Durham University
Durham, UK
jakub.oprsal@durham.ac.uk

ABSTRACT

The complexity and approximability of the constraint satisfaction problem (CSP) has been actively studied over the last 20 years. A new version of the CSP, the promise CSP (PCSP), has recently been proposed, motivated by open questions about the approximability of variants of satisfiability and graph colouring. The PCSP significantly extends the standard decision CSP. The complexity of CSPs with a fixed constraint language on a finite domain has recently been fully classified, greatly guided by the algebraic approach, which uses polymorphisms — high-dimensional symmetries of solution spaces — to analyse the complexity of problems. The corresponding classification for PCSPs is wide open and includes some long-standing open questions, such as the complexity of approximate graph colouring, as special cases.

The basic algebraic approach to PCSP was initiated by Brakensiek and Guruswami, and in this paper we significantly extend it and lift it from concrete properties of polymorphisms to their abstract properties. We introduce a new class of problems that can be viewed as algebraic versions of the (Gap) Label Cover problem, and show that every PCSP with a fixed constraint language is equivalent to a problem of this form. This allows us to identify a "measure of symmetry" that is well suited for comparing and relating the complexity of different PCSPs via the algebraic approach. We demonstrate how our theory can be applied by improving the state-of-the-art in approximate graph colouring: we show that, for any $k \geq 3$, it is NP-hard to find a $(2k-1)$-colouring of a given k-colourable graph.

CCS CONCEPTS

• **Theory of computation → Problems, reductions and completeness**.

KEYWORDS

constraint satisfaction, promise problem, approximation, graph colouring, polymorphism

*The first author was supported by the Austrian Science Fund project P29931, the Czech Science Foundation project 18-20123S, Charles University Research Centre program UNCE/SCI/022 and PRIMUS/SCI/12. The second and the last authors were supported by the UK EPSRC grant EP/R034516/1. The last author was also supported by the European Research Council (Grant Agreement no. 681988, CSP-Infinity).

STOC '19, June 23–26, 2019, Phoenix, AZ, USA

ACM ISBN 978-1-4503-6705-9/19/06.
https://doi.org/10.1145/3313276.3316300

ACM Reference Format:
Jakub Bulín, Andrei Krokhin, and Jakub Opršal. 2019. Algebraic Approach to Promise Constraint Satisfaction. In *Proceedings of the 51st Annual ACM SIGACT Symposium on the Theory of Computing (STOC '19), June 23–26, 2019, Phoenix, AZ, USA.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3313276.3316300

1 INTRODUCTION

What kind of inherent mathematical structure makes a computational problem tractable, i.e., polynomial time solvable (assuming P ≠ NP)? Finding a general answer to this question is one of the fundamental goals of theoretical computer science. The *constraint satisfaction problem (CSP)* and its variants are extensively used towards this ambitious goal for two reasons: on the one hand, the CSP framework is very general and includes a wide variety of computational problems, and on the other hand, this framework has very rich mathematical structure providing an excellent laboratory both for complexity classification methods and for algorithmic techniques.

The basic aim in a CSP is to decide whether there is an assignment of values from some domain A to a given set of variables, subject to constraints on the combinations of values which can be assigned simultaneously to certain specified subsets of variables. There are many variants of this framework (see surveys in [42]). Since the basic CSP is NP-complete (and, for other variants, as hard as it can be) in full generality, a major line of research in the CSP focuses on identifying tractable cases and understanding the mathematical structure enabling tractability (see [42]).

One particular family of CSPs that receives a great amount of attention consists of the CSPs with a fixed *constraint language* [29, 42], i.e., with a restricted set of types of constraints. Since constraints are usually given by relations, a constraint language is simply a set Γ of relations on a domain A. The restricted CSP where only relations from Γ can specify constraints is denoted by CSP(Γ). Many computational problems, including various versions of logical satisfiability, graph colouring, and systems of equations can be represented in this form [29, 42]. It is well-known [29] that the basic CSP can be cast as a homomorphism problem from one relational structure to another (the latter is often called a *template*), and we will use this view. Problems CSP(Γ) correspond to the case when the template structure is fixed. There is an active line of research into CSPs with infinite A (see, e.g. survey [11]), but throughout this paper we assume that A is finite (unless specified otherwise).

In [29], Feder and Vardi conjectured that, for each finite constraint language Γ, the (decision) problem CSP(Γ) is either in P or NP-complete. This conjecture inspired a very active research programme in the last 20 years, which recently culminated in two independent proofs of the conjecture, one by Bulatov [21] and the

other by Zhuk [48] (along with similar classification results for other CSP variants, e.g. [6, 20, 41, 47]). All of these proofs heavily use the so-called algebraic approach to the CSP. On a very high level, this approach uses multivariate functions that preserve relations in a constraint language (and hence solution sets of problem instances), called *polymorphisms*. Thus polymorphisms can be seen as high-dimensional "symmetries" of solution sets. Roughly, lack of such symmetries implies hardness of the corresponding problem, while presence of symmetries implies tractability. This approach was started in a series of papers by Jeavons et al., e.g. [37], where the key role of polymorphisms was established. It was then taken to a more abstract level in [18, 19], where an abstract view on polymorphisms was used, through universal algebras and varieties formed by algebras — this allowed a powerful machinery of structural universal algebra to be applied to the CSP. Another important general methodological improvement was [8], where it was shown that special equations of simple form satisfied by polymorphisms govern the complexity of CSPs. Even though [8] did not impact specifically on the resolution of the Feder-Vardi conjecture, it strongly influenced the present paper.

A new extended version of the CSP, the so-called *promise constraint satisfaction problem* (*PCSP*), has recently been introduced [3, 15], motivated by open problems about (in)approximability for variants of SAT and graph colouring. Roughly, this line of research in approximability concerns finding an approximately good solution to an instance of a (typically hard) problem when a good solution is guaranteed to exist (see discussion and references in [14]). Approximation can be understood in terms of relaxing constraints, or in terms of counting satisfied/violated constraints — in this paper, we use the former. Specifically, in the PCSP, each constraint in an instance has two relations: a 'strict' one, and a 'relaxed' one, and one needs to distinguish between the case when an instance has a solution subject to the strict constraints and the case when it has no solution even subject to the relaxed constraints. One example of such a problem (beyond CSPs) is the case when the only available strict relation is the disequality on a k-element set and the corresponding relaxed relation is the disequality on a c-element set (with $c \geq k$) — the problem is then to distinguish k-colourable graphs from those that are not even c-colourable. This problem (and hence the problem of colouring a given k-colourable graph with c colours) has been conjectured NP-hard, but the question in full generality is still open after more than 40 years of research. We give more examples later. Note that if the strict form and the relaxed form for each constraint coincide, then one gets the standard CSP, so the PCSP framework greatly generalises the CSP.

The problem of systematically investigating the complexity of PCSPs (with a fixed constraint language) was suggested in [3, 15]. We remark that, beyond CSPs, the current knowledge of the complexity landscape of PCSPs is quite limited, and we do not even have analogues of full classification results for graph homomorphisms [34] and Boolean CSPs [46] — which were the most important basic special cases of CSP complexity classifications that inspired the Feder-Vardi conjecture. The quest of complexity classification of PCSPs is of great interest for a number of reasons. It brings together two very advanced methodologies: analysing the complexity of CSPs via algebra and the approximability of CSPs via PCP-based methodology, hence the possibility of fruitful cross-fertilisation and influence beyond the broad CSP framework. It is perfect for further exploring the thesis that (high-dimensional) symmetries of solution spaces are relevant for complexity — which is certainly true for most CSP-related problems, but may be applicable in a wider context. Finally, this quest includes long-standing open problems as special cases.

Related Work. An accessible exposition of the algebraic approach to the CSP can be found in [7], where many ideas and results leading to (but not including) the resolution [21, 48] of the Feder-Vardi conjecture are presented. The volume [42] contains surveys about many aspects of the complexity and approximability of CSPs.

The first link between the algebraic approach and PCSPs was found by Austrin, Håstad, and Guruswami [3], and it was further developed by Brakensiek and Guruswami [13, 15, 16]. They use a notion of polymorphism suitable for PCSPs to prove several hardness and tractability results. Roughly, the polymorphisms of a PCSP (template) are multivariate functions from the domain of its 'strict' relations to that of its 'relaxed' relations that map each strict relation into the corresponding relaxed relation. For example, the n-ary polymorphisms of the PCSP template corresponding to k vs. c graph colouring (we say polymorphisms from $\mathbf{K}_k$ to $\mathbf{K}_c$) are the homomorphisms from the n-th direct power of $\mathbf{K}_k$ to $\mathbf{K}_c$, i.e., the c-colourings of $\mathbf{K}_k^n$. It is shown in [16] that the complexity of a PCSP is fully determined by its polymorphisms — in the sense that two PCSPs with the same set of polymorphisms have the same complexity.

Much of the previous work on the complexity of PCSPs was focused on specific problems, especially on approximate graph and hypergraph colouring and their variants. We describe this in more detail in Examples 2.7–2.12 in the next section. Let us note here that, despite much effort, there is a huge gap between known algorithmic and NP-hardness results for colouring 3-colourable graphs with c colours: the best known NP-hardness result (without additional assumptions) goes only as far as $c = 4$ [31, 39], while the best (in terms of c) known efficient algorithm uses roughly $O(n^{0.199})$ colours to colour an n-vertex 3-colourable graph [38].

We remark that appropriate versions of polymorphisms have been used extensively in many CSP complexity/approximability classifications: standard polymorphisms for decision and counting CSPs, for approximating Min CSPs and testing solutions (in the sense of property testing) [6, 7, 20, 21, 24–26, 48], fractional polymorphisms for exact optimisation problems [41, 47], α-approximate polymorphisms for approximating Max CSPs [17]. In all cases, the presence of nice enough polymorphisms (of appropriate kind) leads directly to efficient algorithms, while their absence leads to hardness. Interestingly, it was shown in [17] that the Unique Games Conjecture is equivalent to the NP-hardness of approximating Max CSPs beyond a specific numerical parameter of their (nice enough) approximate polymorphisms.

Our Contribution. The main contribution of the present paper is a new abstract algebraic theory for the PCSP. A crucial property of polymorphisms for PCSPs is that, unlike in CSPs, they cannot be composed (as functions). The ability to compose polymorphisms to produce new polymorphisms was used extensively in the algebraic theory of CSPs. This could be viewed as a serious limitation on the applicability of the algebraic approach to PCSPs. Alternatively, it is

possible that the ability to compose is not that essential, and that a composition-free abstract algebraic theory for PCSPs (and hence for CSPs) can be developed. Our results suggest that the latter is in fact the case.

We show that certain abstract properties of polymorphisms, namely systems of minor identities (i.e., function equations of a simple form) satisfied by polymorphisms, fully determine the complexity of a PCSP. This shifts the focus from concrete properties of polymorphisms to their abstract properties. Systems of minor identities satisfied by polymorphisms provide a useful measure of how much symmetry a problem has. This measure gives a new tool to compare/relate the complexity of PCSPs, far beyond what was available before. We envisage that our paper will bring a step change in the study of PCSPs, similar to what [18, 19] did for the CSP. Let us explain this in some more detail.

To be slightly more technical, a *minor identity* is a formal expression of the form

$$f(x_1, \dots, x_n) \approx g(x_{\pi(1)}, \dots, x_{\pi(m)})$$

where f, g are function symbols (of arity n and m, respectively), $x_1, \dots, x_n$ are variables, and $\pi\colon [m] \to [n]$ (we use notation $[n] = \{1, \dots, n\}$ throughout). A minor identity can be seen as an equation where the function symbols are the unknowns, and if some specific functions f and g satisfy such an identity then f is called a *minor* of g. We use the symbol $\approx$ instead of $=$ to stress the difference between a formal identity (i.e. equation involving function symbols) and equality of two specific functions. A *minor condition* is a finite system of minor identities (where the same function symbol can appear in several identities). A bipartite minor condition is one where sets of function symbols appearing on the left- and right-hand sides of the identities are disjoint. Such condition is said to be satisfied in a set $\mathscr{F}$ of functions if it is possible to assign a function from $\mathscr{F}$ of the corresponding arity to each of the function symbols in such a way that all the identities are simultaneously satisfied (as equalities of functions, i.e., for all possible values of the x_i's).

Informally, the main results of our new general theory state that

(A) If every bipartite minor condition satisfied in the polymorphisms of (the template of) one PCSP Π_1 is also satisfied in the polymorphisms of another PCSP Π_2, then Π_2 is log-space reducible to Π_1 (see Theorem 3.1 for a formal statement).

(B) Every PCSP Π is log-space equivalent to the problem deciding whether a given bipartite minor condition is satisfiable by projections/dictators or not satisfiable even by polymorphisms of Π (see Theorem 3.9).

The first of the above results establishes the key role of bipartite minor conditions satisfied in polymorphisms — in particular, the hardness or tractability of a PCSP can always be explained on this abstract level, since any two PCSPs have the same complexity if their polymorphisms satisfy the same bipartite minor conditions. Moreover, this abstract level allows one to compare any two PCSPs, even when it does not makes sense to compare their sets of polymorphisms inclusion-wise (say, because the functions involved are defined on different sets). The second result establishes that every PCSP is equivalent to what can be viewed as an algebraic version of the Gap Label Cover problem, which is the most common starting point of PCP-based hardness proofs in the inapproximability context. Our result uses the fact the Label Cover can be naturally interpreted as the problem, which we call MC, of checking triviality of a system of minor identities. The gap version of MC has an algebraic component in place of the quantitative gap of Gap Label Cover. In particular, result (B) can provide a general approach to proving NP-hardness of PCSPs — via analysis of bipartite minor conditions satisfied by polymorphisms. The full version of our general theory that extends beyond the above results can be found in [22].

As a first application of our general theory, we prove the following result.

(C) For any $k \geq 3$, it is NP-hard to distinguish k-colourable graphs from those that are not $(2k-1)$-colourable. (See Theorem 4.1).

In particular, it follows that it is NP-hard to 5-colour a 3-colourable graph. This might seem a small step towards closing the big gap in our understanding of approximate graph colouring, but we believe that it is important methodologically and that further development of our general theory and further analysis of polymorphisms for graph colouring will eventually lead to a proof of NP-hardness for any constant number of colours.

For better accessibility, we will give here a direct (polymorphism-based) proof of result (C), but in the full version [22] we show that this case of approximate graph colouring has less symmetry (in the sense of bipartite minor conditions) than approximate hypergraph colouring, which is known to be NP-hard [28]. Then our theory implies the required reduction. Our theory also allows one to rule out the existence of certain reductions — for example, we can explain exactly how k vs. $c = 2k-1$ colouring differs from the cases $c = 2k-2$ and $c = 2k$, and hence why we are able to improve the result from $c = 2k-2$ [13] to $c = 2k-1$ and why moving to larger c requires further analysis of polymorphisms and bipartite minor conditions. For this and a number of further applications, see [22].

Overview of Key Technical Ideas. At the heart of our theory is a new class of problems, the problems of deciding triviality of minor conditions, and their promise versions that can be viewed as algebraic counterparts of the Gap Label Cover problem. Recall that an instance of the Label Cover problem (LC) is given by a bipartite graph $G = (U, V; E)$ whose nodes are viewed as variables, domains $[m]$ and $[n]$ for the variables in U and V, respectively, and a map $\pi_e\colon [m] \to [n]$ for each $e \in E$. The Label Cover problem is a CSP where the constraints are given by the maps π_e, and such a constraint $\pi_{(u,v)}$ is satisfied by $u = i$ and $v = j$ if $\pi(j) = i$. We interpret such a constraint as the minor identity

$$f_u(x_1, \dots, x_n) \approx g_v(x_{\pi(1)}, \dots, x_{\pi(m)}).$$

This identity is satisfied by assigning the projection (dictator) on the i-th coordinate to f_u and the projection on the j-th coordinate to g_v if and only if (i, j) satisfies the corresponding Label Cover constraint on (u, v). Therefore, the problem (which we call MC) of deciding whether a given bipartite minor condition is trivial, i.e., can be satisfied in projections, is just a different interpretation of Label Cover.

The key advantage of MC over LC is that it allows us to define a promise version of the problem, that we call PMC (for 'promise MC'), that is more suitable for our setting than the Gap Label Cover. Whereas the Gap Label Cover problem (with perfect completeness

and soundness ε) asks to distinguish fully satisfiable LC instances from those where at most ε-fraction of constraints can be satisfied, the PMC problem asks to distinguish trivial systems of bipartite minor conditions from those that cannot be satisfied even in some fixed family $\mathscr{F}$ of multivariate functions (from one fixed set to another). We denote the latter problem by $\mathrm{PMC}_{\mathscr{F}}$. This problem is well-defined for any family $\mathscr{F}$ that is a *minion*, i.e., closed under taking minors, since satisfiability in projections implies satisfiability in any minion.

We prove that every PCSP is log-space equivalent to the problem $\mathrm{PMC}_{\mathscr{M}}$ where $\mathscr{M}$ is the minion of all polymorphisms of this PCSP (this is our result (B)). For this, we adapt reductions from CSP to LC and back to our setting. We build on [1, 10] to provide a reduction from a PCSP to PMC, and on [9, 37] for a reduction from PMC to a PCSP. Our result (A) is proved by using the above reductions, i.e., by first reducing PCSP Π_2 to the corresponding PMC problem, then reducing that to the PMC corresponding to PCSP Π_1 and finally by reducing this PMC to PCSP Π_1. For the middle reduction to work, we use the assumption that every (bipartite) minor condition satisfied by polymorphisms of Π_1 is also satisfied by polymorphisms of Π_2 (to align the NO-instances of the two PMCs). This assumption is formally captured by the existence of a minor-preserving mapping from the polymorphisms of Π_1 to the polymorphisms of Π_2, which is called a *minion homomorphism* (extending a similar notion from [8]). We remark that proving the existence of a minion homomorphism for specific cases can be a very non-trivial problem.

Organisation of This Paper. The following section contains formal definitions of the concepts mentioned in the introduction as well as several concrete examples of PCSPs. Section 3 introduces reductions via algebraic conditions and proves the main results of our general theory. Section 4 contains the proof of new NP-hardness results for approximate graph colouring. Finally, Section 5 summarises the results of the paper and discusses possible directions of further research.

2 PRELIMINARIES

This section contains formal definitions of the notions introduced above. For comparison, the algebraic theory behind fixed-template CSPs can be found in a recent survey [7].

2.1 CSPs and PCSPs

We use the notation $[n] = \{1, \dots, n\}$ and $E_n = \{0, 1, \dots, n-1\}$ throughout the paper.

Definition 2.1. A *constraint language* Γ on a set A is finite set of relations on A, possibly of different arity. Then A is called the *domain* of Γ.

To work with several constraint languages (possibly on different domains), it is often convenient to fix an indexing of the relations in Γ. This is formalised as follows.

Definition 2.2. A *(relational) structure* with domain A is a tuple $\mathbf{A} = (A; R_1^{\mathbf{A}}, \dots, R_n^{\mathbf{A}})$ where each $R_i^{\mathbf{A}} \subseteq A^{\mathrm{ar}(R_i)}$ is a relation on A of arity $\mathrm{ar}(R_i) \geq 1$. We say that $\mathbf{A}$ is finite if A is finite. We will assume that all structures in this paper are finite unless specified otherwise.

Two structures $\mathbf{A} = (A; R_1^{\mathbf{A}}, \dots, R_n^{\mathbf{A}})$ and $\mathbf{B} = (B; R_1^{\mathbf{B}}, \dots, R_n^{\mathbf{B}})$ are called *similar* if they have the same number of relations and $\mathrm{ar}(R_i^{\mathbf{A}}) = \mathrm{ar}(R_i^{\mathbf{B}})$ for each $i \in [n]$.

For example, a (directed) graph is relational structure with one binary relation. Any two graphs are similar structures.

We often use a single letter instead of R_i to denote a relation of a structure, e.g. $S^{\mathbf{A}}$ would denote a relation of $\mathbf{A}$, the corresponding relation in a similar structure $\mathbf{B}$, would be denoted by $S^{\mathbf{B}}$. Also, throughout the paper we denote the domains of structures $\mathbf{A}$, $\mathbf{B}$, $\mathbf{K}_n$ and so on by A, B, K_n etc., respectively.

Definition 2.3. For two similar relational structures $\mathbf{A}$ and $\mathbf{B}$, a *homomorphism* from $\mathbf{A}$ to $\mathbf{B}$ is a map $h\colon A \to B$ such that, for each i,

$$\text{if } (a_1, \dots, a_{\mathrm{ar}(R_i)}) \in R_i^{\mathbf{A}} \text{ then } (h(a_1), \dots, h(a_{\mathrm{ar}(R_i)})) \in R_i^{\mathbf{B}}.$$

We write $h\colon \mathbf{A} \to \mathbf{B}$ to denote this, and simply $\mathbf{A} \to \mathbf{B}$ to denote that a homomorphism from $\mathbf{A}$ to $\mathbf{B}$ exists. In the latter case, we also say that $\mathbf{A}$ maps homomorphically to $\mathbf{B}$.

Definition 2.4. For a fixed structure $\mathbf{B}$, $\mathrm{CSP}(\mathbf{B})$ is the problem of deciding whether a given input structure $\mathbf{I}$, similar to $\mathbf{B}$, admits a homomorphism to $\mathbf{B}$. In this case $\mathbf{B}$ is called the *template* for $\mathrm{CSP}(\mathbf{B})$.

For example, when $\mathbf{I}$ and $\mathbf{B}$ are graphs and $\mathbf{B} = \mathbf{K}_k$ is a k-clique, a homomorphism from $\mathbf{I}$ to $\mathbf{B}$ is simply a k-colouring of $\mathbf{I}$. Then $\mathrm{CSP}(\mathbf{B})$ is the standard k-colouring problem for graphs.

To see how the above definition corresponds to the definition of CSP with variables and constraints, view the domain of the structure $\mathbf{I}$ as consisting of variables, relations in $\mathbf{I}$ specifying which tuples of variables the constraints should be applied to, and (the corresponding) relations in $\mathbf{B}$ as sets of allowed tuples of values.

Definition 2.5. A *PCSP template* is a pair of similar structures $\mathbf{A}$ and $\mathbf{B}$ such that $\mathbf{A} \to \mathbf{B}$. The problem $\mathrm{PCSP}(\mathbf{A}, \mathbf{B})$ is, given an input structure $\mathbf{I}$ similar to $\mathbf{A}$ and $\mathbf{B}$, output YES if $\mathbf{I} \to \mathbf{A}$, and NO if $\mathbf{I} \nrightarrow \mathbf{B}$.

Note that $\mathrm{PCSP}(\mathbf{A}, \mathbf{A})$ is simply $\mathrm{CSP}(\mathbf{A})$. The promise in the PCSP is that it is never the case that $\mathbf{I} \nrightarrow \mathbf{A}$ and $\mathbf{I} \to \mathbf{B}$. Note also that the assumption $\mathbf{A} \to \mathbf{B}$ is necessary for the problem to make sense — otherwise, the YES- and NO-cases would not be disjoint. We define $\mathrm{PCSP}(\mathbf{A}, \mathbf{B})$ as a decision problem, but it can also be defined as a search problem:

Definition 2.6. Given two relational structures $\mathbf{A}$, $\mathbf{B}$ as above, the search version of $\mathrm{PCSP}(\mathbf{A}, \mathbf{B})$ is, given an input structure $\mathbf{I}$ that maps homomorphically to $\mathbf{A}$, *find* a homomorphism $h\colon \mathbf{I} \to \mathbf{B}$.

There is an obvious reduction from the decision variant of every PCSP to its search variant. Nevertheless, it is open whether these two problems are equivalent for all PCSP templates. Note that, for problems $\mathrm{CSP}(\mathbf{A})$, these two versions are always equivalent [19]. We would also like to note that all of our results are formulated and proved for the decision version. Nevertheless, all of them can be generalised to the corresponding search versions of the problems.

Let us give some examples of the problems of the form $\mathrm{PCSP}(\mathbf{A}, \mathbf{B})$ which are proper promise problems, i.e., not of the form $\mathrm{CSP}(\mathbf{A})$. More examples (of both tractable and intractable PCSPs) can be found in [13, 15, 16].

Example 2.7 ($(2+\varepsilon)$-Sat). For a tuple $\mathbf{t} \in \{0,1\}^n$, let $\mathrm{Ham}(\mathbf{t})$ be the Hamming weight of $\mathbf{t}$. Fix an integer $k \geq 1$. Let $\neq_2$ be the relation $\{(0,1),(1,0)\}$. Let

$$\mathbf{A} = (\{0,1\}; \{\mathbf{t} \in \{0,1\}^{2k+1} \mid \mathrm{Ham}(\mathbf{t}) \geq k\}, \neq_2),$$

$$\mathbf{B} = (\{0,1\}; \{\mathbf{t} \in \{0,1\}^{2k+1} \mid \mathrm{Ham}(\mathbf{t}) \geq 1\}, \neq_2).$$

The problem $\mathrm{PCSP}(\mathbf{A},\mathbf{B})$ is then (equivalent to) the following variant of $(2k+1)$-Sat: given an instance of $(2k+1)$-Sat such that some assignment satisfies at least k literals in each clause, find a normal satisfying assignment. This problem, called $(2+\varepsilon)$-Sat, was proved to be NP-hard in [3].

Example 2.8 (1-in-3- vs. Not-All-Equal-Sat). Let

$$\mathbf{T} = (\{0,1\}; \{(1,0,0),(0,1,0),(0,0,1)\}),$$

$$\mathbf{H}_2 = (\{0,1\}; \{0,1\}^3 \setminus \{(0,0,0),(1,1,1)\}).$$

Even though $\mathrm{CSP}(\mathbf{T})$ and $\mathrm{CSP}(\mathbf{H}_2)$ are (well-known) NP-hard problems, the problem $\mathrm{PCSP}(\mathbf{T},\mathbf{H}_2)$ was shown to be in P in [15, 16], along with a range of similar problems.

Example 2.9 (Approximate Graph Colouring). For $k \geq 2$, let $\neq_k$ denote the relation $\{(a,b) \in E_k^2 \mid a \neq b\}$. For $k \leq c$, let

$$\mathbf{K}_k = (E_k; \neq_k),$$

$$\mathbf{K}_c = (E_c; \neq_c).$$

Then $\mathrm{PCSP}(\mathbf{K}_k,\mathbf{K}_c)$ is the well-known *approximate graph colouring* problem: given a k-colourable graph, find a c-colouring. The decision version of this asks to distinguish between k-colourable graphs and those that are not even c-colourable. The complexity of this problem has been studied since 1976 [30] — the problem has been conjectured to be NP-hard for any fixed $3 \leq k \leq c$, but this is still open in many cases (see [13, 31, 35, 39, 40]) if we assume only $\mathrm{P} \neq \mathrm{NP}$. For $k = 3$, the case $c = 4$ was shown to be NP-hard [31, 39], but even the case $c = 5$ was still open, and we settle it in this paper. It was shown (using polymorphisms) in [13] that, for any $k \geq 3$, it is NP-hard to distinguish k-colourable graphs from those that are not $(2k-2)$-colourable. This gives the best known NP-hardness results for small enough k, but we further improve this result in this paper. For large enough k, the best known NP-hardness result is for k vs. $2^{\Omega(k^{1/3})}$ colouring [35]. By additionally assuming somewhat non-standard variants of the Unique Games Conjecture (with perfect completeness), NP-hardness of all approximate graph colouring problems (with $k \geq 3$) was proved in [27]. We note that an extension of this problem to graph homomorphisms (in the spirit of [34]) was proposed in [16].

Example 2.10 (Approximate Hypergraph Colouring). This problem is similar to the previous one, but uses the "not-all-equal" relation $\mathrm{NAE}_k = E_k^3 \setminus \{(a,a,a) \mid a \in E_k\}$ instead of $\neq_k$, and similarly for c, i.e., we are talking about $\mathrm{PCSP}(\mathbf{H}_k,\mathbf{H}_c)$ where

$$\mathbf{H}_k = (E_k; \mathrm{NAE}_k),$$

$$\mathbf{H}_c = (E_c; \mathrm{NAE}_c).$$

A colouring of a hypergraph is an assignment of colours to its vertices that leaves no hyperedge monochromatic. Thus, in (the search variant of) this problem one needs to find a c-colouring for a given k-colourable 3-uniform hypergraph. This problem has been proved to be NP-hard for any fixed $2 \leq k \leq c$ [28].

Example 2.11 (Strong vs. Normal Hypergraph Colouring). Let $k, c \geq 2$, and let

$$\mathbf{A} = (E_{k+1}; \{(a_1,\ldots,a_k) \in E_{k+1}^k \mid a_i \neq a_j \text{ for all } i,j\}),$$

$$\mathbf{B} = (E_c; \{(a_1,\ldots,a_k) \in E_c^k \mid a_i \neq a_j \text{ for some } i,j\}).$$

Then $\mathrm{PCSP}(\mathbf{A},\mathbf{B})$ is the problem of distinguishing k-uniform hypergraphs that admit a strong $(k+1)$-colouring (i.e., one without repetition of colours in any hyperedge) from those that do not admit a normal c-colouring. It was conjectured in [13] that this problem is NP-hard for all $(k,c) \neq (2,2)$, and some special cases ($k = 3,4$ and $c = 2$) were settled in that paper, but this conjecture remains wide open.

Example 2.12 (Rainbow vs. Normal Hypergraph Colouring). Let k, q, c be positive integers with $k \geq q \geq 2$ and $c \geq 2$, and let

$$\mathbf{R}_{k,q} = (E_q; \{(a_1,\ldots,a_k) \in E_q^k \mid \{a_1,\ldots,a_k\} = E_q\}),$$

$$\mathbf{H}_{k,c} = (E_c; \{(a_1,\ldots,a_k) \in E_c^k \mid a_i \neq a_j \text{ for some } i,j\}).$$

In $\mathrm{PCSP}(\mathbf{R}_{k,q},\mathbf{H}_{k,c})$, one is given a k-uniform hypergraph which has a q-colouring such that all colours appear in each hyperedge, and one needs to find a normal c-colouring. This problem is known to be in P for $k = q$ and $c = 2$; a randomised algorithm for it can be found in [43], and a deterministic algorithm due to Alon (unpublished) is mentioned in [15]. $\mathrm{PCSP}(\mathbf{R}_{k,q},\mathbf{H}_{k,c})$ is NP-hard if $2 \leq q \leq \lfloor k/2 \rfloor$ [32] or if $2 \leq q \leq k - 2\lfloor \sqrt{k} \rfloor$ and $c = 2$ [2]. Further variations of such PCSPs were considered in [2, 32].

2.2 Polymorphisms

We now proceed to define polymorphisms, which are the main algebraic technical tools used in the analysis of CSPs.

Definition 2.13. The (direct) *n-th power* of $\mathbf{A}$ is the structure $\mathbf{A}^n = (A^n; R_1^{\mathbf{A}^n}, \ldots, R_n^{\mathbf{A}^n})$ whose relations are defined as follows: for every $\mathrm{ar}(R_i) \times n$ matrix M such that all columns of M are in $R_i^{\mathbf{A}}$, consider the rows of M as elements of A^n and put this tuple in $R_i^{\mathbf{A}^n}$.

Definition 2.14. Given two similar relational structures $\mathbf{A}$ and $\mathbf{B}$, an *n-ary polymorphism*[1] *from* $\mathbf{A}$ *to* $\mathbf{B}$ is a homomorphism from $\mathbf{A}^n$ to $\mathbf{B}$. To spell this out, a polymorphism is a mapping f from A^n to B such that, for each $i \leq n$ and all tuples $(a_{11},\ldots,a_{\mathrm{ar}(R_i)1}), \ldots, (a_{1n},\ldots,a_{\mathrm{ar}(R_i)n}) \in R_i^{\mathbf{A}}$, we have

$$(f(a_{11},\ldots,a_{1n}),\ldots,f(a_{\mathrm{ar}(R_i)1},\ldots,a_{\mathrm{ar}(R_i)n})) \in R_i^{\mathbf{B}}.$$

We denote the set of all polymorphisms from $\mathbf{A}$ to $\mathbf{B}$ by $\mathrm{Pol}(\mathbf{A},\mathbf{B})$, and we write simply $\mathrm{Pol}(\mathbf{A})$ for $\mathrm{Pol}(\mathbf{A},\mathbf{A})$.

For the case $\mathbf{A} = \mathbf{B}$, this definition coincides with the standard definition of a polymorphism of a relational structure $\mathbf{A}$ (see, e.g., [7, Section 4]).

Definition 2.15. A *projection* (also known as *dictator*[2]) on a set A is an operation $p_i^{(n)} \colon A^n \to A$ of the form $p_i^{(n)}(x_1,\ldots,x_n) = x_i$.

[1] Called *weak polymorphism* in [3, 13].

[2] Projection is the standard name for these objects in universal algebra, while dictator is the standard name in approximability literature.

It is well-known and easy to see that every projection on A is a polymorphism of every relational structure on A. We will sometimes write simply p_i when the arity n of a projection is clear from the context.

Example 2.16. Consider the structures $\mathbf{T}$ and $\mathbf{H}_2$ from Example 2.8. It is well-known and not hard to verify that

- $\mathrm{Pol}(\mathbf{T})$ consists of all projections on $\{0,1\}$, and
- $\mathrm{Pol}(\mathbf{H}_2)$ consists of all operations of the form $\pi(p_i^{(n)})$ where π is a permutation on $\{0,1\}$ and $p_i^{(n)}$ is a projection.

However, $\mathrm{Pol}(\mathbf{T},\mathbf{H}_2)$ contains many operations that are not like projections. For example, for any $k \geq 1$, let $f_k\colon \{0,1\}^{3k-1} \to \{0,1\}$ be such that $f_k(\mathbf{t}) = 1$ if $\mathrm{Ham}(\mathbf{t}) \geq k$ and $f_k(\mathbf{t}) = 0$ otherwise. It is easy to see that $f_k \in \mathrm{Pol}(\mathbf{T},\mathbf{H}_2)$. Indeed, if M is a $3\times(3k-1)$ matrix whose columns come from the set $\{(1,0,0),(0,1,0),(0,0,1)\}$ then some row of M contains strictly fewer than k 1's and some other row contains at least k 1's. Therefore, by applying f_k to the rows of M, one obtains a tuple in $\{0,1\}^3 \setminus \{(0,0,0),(1,1,1)\}$, as required.

Example 2.17. Recall Example 2.9. It is easy to check that, for any $n \geq 1$, the n-ary functions from $\mathrm{Pol}(\mathbf{K}_k,\mathbf{K}_c)$ are simply the c-colourings of $\mathbf{K}_k^n$, the n-th power of $\mathbf{K}_k$ (in the sense of Definition 2.13).

Brakensiek and Guruswami showed that, when the pair of domains for PCSP templates is fixed, the polymorphisms determine the complexity of a PCSP.

Theorem 2.18 ([13]). *Let* $(\mathbf{A},\mathbf{B})$ *and* $(\mathbf{A}',\mathbf{B}')$ *be a pair of PCSP templates over the same pair of domains. If* $\mathrm{Pol}(\mathbf{A}',\mathbf{B}') \subseteq \mathrm{Pol}(\mathbf{A},\mathbf{B})$, *then PCSP*$(\mathbf{A},\mathbf{B})$ *is polynomial-time reducible to PCSP*$(\mathbf{A}',\mathbf{B}')$.

Unlike polymorphisms of a single relational structure $\mathbf{A}$, the set $\mathrm{Pol}(\mathbf{A},\mathbf{B})$ is not closed under composition — for example, if $f(x,y,z)$ and $g(x,y)$ are in $\mathrm{Pol}(\mathbf{A},\mathbf{B})$, then $f(g(x,w),y,z)$ is not necessarily there. In general, the composition is not always well-defined, and even when it is (e.g. when $\mathbf{A}$ and $\mathbf{B}$ have the same domain), $f(g(x,w),y,z)$ may not be in $\mathrm{Pol}(\mathbf{A},\mathbf{B})$. However, it is always closed under taking minors.

Definition 2.19. An n-ary function $f\colon A^n \to B$ is called a *minor* of an m-ary function $g\colon A^m \to B$ given by a map $\pi\colon [m] \to [n]$ if

$$f(x_1,\dots,x_n) = g(x_{\pi(1)},\dots,x_{\pi(m)})$$

for all $x_1,\dots,x_n \in A$.

Alternatively, one can say that f is a minor of g if it is obtained from g by identifying variables, permuting variables, and introducing dummy variables.

Definition 2.20. Let $\mathscr{O}(A,B) = \{f\colon A^n \to B \mid n \geq 1\}$. A *(function) minion* $\mathscr{M}$ on a pair of sets (A,B) is a non-empty subset of $\mathscr{O}(A,B)$ that is closed under taking minors. For fixed $n \geq 1$, let $\mathscr{M}^{(n)}$ denote the set of n-ary functions from $\mathscr{M}$.

We remark that clones have been used extensively in the algebraic theory of CSP — a clone is simply a minion on (A,A) that is closed under composition and contains all projections. For any structure $\mathbf{A}$, $\mathrm{Pol}(\mathbf{A})$ is a clone. For more detail, see [7].

We now introduce one of the central notions of this paper; it generalises the notion of *h1 clone homomorphisms* from [8].

Definition 2.21. Let $\mathscr{M}$ and $\mathscr{N}$ be two minions (not necessarily on the same pairs of sets). A mapping $\xi\colon \mathscr{M} \to \mathscr{N}$ is called a *minion homomorphism* if

(1) it preserves arities, i.e., $\mathrm{ar}(g) = \mathrm{ar}(\xi(g))$ for all $g \in \mathscr{M}$, and
(2) it preserves taking minors, i.e., for each $\pi\colon [m] \to [n]$ and each $g \in \mathscr{M}^{(m)}$ we have

$$\xi(g)(x_{\pi(1)},\dots,x_{\pi(m)}) = \xi(g(x_{\pi(1)},\dots,x_{\pi(m)})).$$

Item (2) above can also be interpreted as 'preserving satisfaction of minor identities', i.e., if $f(x_1,\dots,x_n) = g(x_{\pi(1)},\dots,x_{\pi(m)})$ for some $f \in \mathscr{M}^{(n)}$, $g \in \mathscr{M}^{(m)}$, and $\pi\colon [m] \to [n]$, then

$$\xi(f)(x_1,\dots,x_n) = \xi(g)(x_{\pi(1)},\dots,x_{\pi(m)}).$$

Example 2.22. We will construct a minion homomorphism from $\mathrm{Pol}(\mathbf{K}_3,\mathbf{K}_4)$ to the minion $\mathscr{P}_2$ of all projections on a two-element set.

Our minion homomorphism is built on the following combinatorial statement proved in [13, Lemma 3.4]: for each $f \in \mathrm{Pol}^{(n)}(\mathbf{K}_3,\mathbf{K}_4)$, there exist $t \in K_4$ (we will call any such t a *trash colour*), a coordinate $i \in [n]$, and a map $\alpha\colon K_3 \to K_4$ such that

$$f(a_1,\dots,a_n) \in \{t,\alpha(a_i)\}$$

for all $a_1,\dots,a_n \in K_3$. In other words, if we erase the value t from the table of f then $f(x_1,\dots,x_n)$, which is now a partial function, depends only on x_i. Moreover, it is shown in [13, Lemma 3.9], that while the trash colour t is not necessarily unique, the coordinate i is.

We define $\xi\colon \mathrm{Pol}(\mathbf{K}_3,\mathbf{K}_4) \to \mathscr{P}_2$ by mapping each f to p_i (preserving the arity) for the i that satisfies the above. To prove that such ξ is a minion homomorphism, consider f,g such that f is a minor of g, i.e.,

$$f(x_1,\dots,x_n) = g(x_{\pi(1)},\dots,x_{\pi(m)})$$

for some $\pi\colon [m] \to [n]$. We claim that if t is a trash colour for g, then it is also a trash colour for f. Indeed, if $g(a_1,\dots,a_m) \in \{t,\alpha(a_i)\}$ for all $a_1,\dots,a_n \in K_3$, we get

$$f(a_1,\dots,a_n) = g(a_{\pi(1)},\dots,a_{\pi(m)}) \in \{t,\alpha(a_{\pi(i)})\}.$$

This also shows that the coordinate j assigned to f is $\pi(i)$, therefore $\xi(f) = p_{\pi(i)}$. We conclude that ξ preserves taking minors. That is true since

$$p_{\pi(i)}(x_1,\dots,x_n) = x_{\pi(i)} = p_i(x_{\pi(1)},\dots,x_{\pi(m)}).$$

3 ALGEBRAIC REDUCTIONS

In this section, we prove the two main results of our general theory. The first one can be formulated right away.

Theorem 3.1. *Let* $(\mathbf{A}_1,\mathbf{B}_1)$ *and* $(\mathbf{A}_2,\mathbf{B}_2)$ *be two finite PCSP templates, and let* $\mathscr{M}_i = \mathrm{Pol}(\mathbf{A}_i,\mathbf{B}_i)$ *for* $i = 1,2$. *If there is a minion homomorphism* $\xi\colon \mathscr{M}_1 \to \mathscr{M}_2$ *then PCSP*$(\mathbf{A}_2,\mathbf{B}_2)$ *is log-space reducible to PCSP*$(\mathbf{A}_1,\mathbf{B}_1)$.

The proof of this theorem is provided by the second of our results, but to formulate it, we need to first address a few formalities about the problem of deciding minor conditions. We remark that, in the full version [22], we characterise the existence of such a minion homomorphism in many equivalent ways.

3.1 Deciding Satisfiability of Bipartite Minor Conditions

Definition 3.2. A *bipartite minor condition* is a finite set Σ of minor identities where the sets of function symbols used on the right- and left-hand sides are disjoint. More precisely, we say that a minor condition Σ is *bipartite* over two disjoint sets of function symbols $\mathcal{U}$ and $\mathcal{V}$, if it contains only identities of the form

$$f(x_1, \dots, x_n) \approx g(x_{\pi(1)}, \dots, x_{\pi(m)})$$

where $f \in \mathcal{U}$ and $g \in \mathcal{V}$ are symbols of arity n and m, respectively, $x_1, \dots, x_n$ are variables, and $\pi\colon [m] \to [n]$.

Such a condition is said to be *satisfied* in a minion $\mathscr{M}$ on (A, B) if there is an assignment $\zeta\colon \mathcal{U} \cup \mathcal{V} \to \mathscr{M}$ that assigns to each function symbol a function from $\mathscr{M}$ of the corresponding arity so that

$$\zeta(f)(a_1, \dots, a_n) = \zeta(g)(a_{\pi(1)}, \dots, a_{\pi(m)})$$

for each identity $f(x_1, \dots, x_n) \approx g(x_{\pi(1)}, \dots, x_{\pi(m)})$ in Σ and all $a_1, \dots, a_n \in A$. We say that a minor condition is *trivial* if it is satisfied in every minion, in particular, in the minion $\mathscr{P}_A$ consisting of all projections (dictators) on a set A that contains at least two elements.

We note that as long as a bipartite minor condition is satisfied in some $\mathscr{P}_A$ with $|A| \geq 2$, it is satisfied in every minion: Recall, that by definition every minion $\mathscr{M}$ is non-empty, and therefore it contains a unary function f (obtained by identifying all variables in a function from $\mathscr{M}$). Consequently, $\mathscr{M}$ contains functions defined by $(x_1, \dots, x_n) \mapsto f(x_i)$ for each i. These functions then behave similarly to projections in $\mathscr{P}_A$.

The symbols f, g, etc. in a minor condition are abstract function symbols. Nevertheless, we sometimes (in particular, when working with specific simple minor conditions) use the same symbols to denote concrete functions that satisfy this condition. When we want to stress the assignment of concrete functions to symbols, we use $\zeta(f)$ for the concrete function assigned to the abstract symbol f.

Example 3.3. Consider the following bipartite minor condition. We set $\mathcal{U}$ to contain a single binary symbol f, and $\mathcal{V}$ a single quaternary symbol g. The set Σ then consists of identities:

$$\begin{aligned} f(x,y) &\approx g(y,x,x,x)\\ f(x,y) &\approx g(x,y,x,x)\\ f(x,y) &\approx g(x,x,y,x)\\ f(x,y) &\approx g(x,x,x,y). \end{aligned}$$

This condition is not trivial, since if $f = p_1$ and $g = p_i$ for some i then the i-th identity is not satisfied, and if $f = p_2$ then the first identity forces that $g = p_1$ which contradicts any of the other identities.

The above bipartite minor condition is satisfied in the minion $\mathrm{Pol}(\mathbf{H}_2, \mathbf{H}_k)$ (recall Example 2.10) for each $k \geq 4$. We define a function $\zeta(g)$ by the following:

$$\zeta(g)(x,y,z,u) = \begin{cases} a & \text{if at least 3 arguments are equal to } a;\\ x+2 & \text{otherwise.} \end{cases}$$

The function $\zeta(f)$ is defined by $\zeta(f)(x,y) = x$. Clearly $\zeta(f)$ and $\zeta(g)$ satisfy the required identities, also $\zeta(f)$ is in $\mathrm{Pol}(\mathbf{H}_2, \mathbf{H}_k)$. We now show that also $\zeta(g)$ is. Consider a 3×4 matrix $M = (a_{ij})$ such that each column of M is a triple in NAE_2. We need to show that applying $\zeta(g)$ to the rows of M gives a triple in NAE_k. For contradiction, assume that we get a triple of the form (a, a, a). If a is 0 or 1, then in each row of M at least three entries are equal to a. In this case, one of columns of M is $(a, a, a) \notin \mathrm{NAE}_2$, a contradiction. Otherwise, a is 2 or 3, which implies that the first column of M is $(a-2, a-2, a-2)$, a contradiction again.

It is easy to show (see [22]) that the above minor condition is also satisfied in $\mathrm{Pol}(\mathbf{K}_3, \mathbf{K}_5)$.

Example 3.4. We now give a simple minor condition that is *not* satisfied in $\mathrm{Pol}(\mathbf{H}_2, \mathbf{H}_k)$ for any $k \geq 2$:

$$\begin{aligned} f(x,y) &\approx g(x,x,y,y,y,x),\\ f(x,y) &\approx g(x,y,x,y,x,y),\\ f(x,y) &\approx g(y,x,x,x,y,y). \end{aligned}$$

Note that the columns of x's and y's on the right above correspond to the triples in NAE_2. Now assume that this condition is satisfied by some $\zeta(f), \zeta(g) \in \mathrm{Pol}(\mathbf{H}_2, \mathbf{H}_k)$, so the identities above become equalities. Then substitute 0 for x and 1 for y in these equalities. The triple (column) on the right-hand side of the system is obtained by applying $\zeta(g)$ to the six triples in NAE_2, so it must be in NAE_k. On the other hand, this triple is equal to (b, b, b) where $b = \zeta(f)(0, 1)$, which is not in NAE_k.

In Section 4 we prove that, for any $k \geq 3$, this minor condition is not satisfied in $\mathrm{Pol}(\mathbf{K}_k, \mathbf{K}_{2k-1})$ either — this is the key part in our proof that $\mathrm{PCSP}(\mathbf{K}_k, \mathbf{K}_{2k-1})$ is NP-hard.

Definition 3.5. We define the problem $\mathrm{MC}(N)$ (*triviality of a bipartite minor condition*) as the problem where the input is a triple $(\Sigma, \mathcal{U}, \mathcal{V})$, with Σ a bipartite minor condition over $\mathcal{U}$ and $\mathcal{V}$ that involves function symbols of maximal arity N, and the goal is to decide whether the condition Σ is trivial.

Deciding triviality of bipartite minor conditions is essentially just a different interpretation of the Label Cover problem that was introduced in [1]. To compare these two problems, we use a formulation of Label Cover that is closer to the one that appeared in e.g. [3] and [13]. In addition, we bound the size of the label sets by a constant N. Some bounded version often appears in the literature as it is well-known that if $N \geq 3$ then it is an NP-complete problem (see, e.g., [13, Lemma 4.4]).

Definition 3.6 (Label cover)**.** Fix a positive integer N. We define $\mathrm{LC}(N)$ as the following decision problem: The input is a tuple (U, V, E, l, r, Π) where

- $G = (U, V; E)$ is a bipartite graph
- $l, r \leq N$ are positive integers, and
- Π is a family of maps $\pi_e\colon [r] \to [l]$, one for each $e \in E$.

The goal is to decide whether there is a labelling of vertices from U and V with labels from $[r]$ and $[l]$, respectively, such that if $(u, v) \in E$ then the label of v is mapped by $\pi_{(u,v)}$ to the label of u.

We interpret a label cover instance (U, V, E, l, r, Π) with $l, r \leq N$ as a bipartite minor condition Σ, an input to $\mathrm{MC}(N)$, as follows.

- Each vertex $u \in U$ is interpreted as an l-ary function symbol f_u, and each vertex $v \in V$ as an r-ary function symbol g_v.
- For each edge $e = (u, v)$ we add to Σ the identity

$$f_u(x_1, \dots, x_l) \approx g_v(x_{\pi_e(1)}, \dots, x_{\pi_e(r)}). \qquad (\heartsuit)$$

Observe that Σ is indeed a bipartite minor condition over $\mathcal{U} = \{f_u \mid u \in U\}$ and $\mathcal{V} = \{g_v \mid v \in V\}$. We claim that the minor condition obtained in this way is trivial if and only if the original Label Cover instance has a solution. The main difference is that a solution to Label Cover is a labelling while a solution (a witness) to triviality of minor conditions is an assignment of projections to the function symbols. Nevertheless, there is a clear bijection between the labels and the projections: label i corresponds to projection p_i. Clearly, a constraint $((u,v),\pi)$ of the Label Cover is satisfied by a pair of labels (i,j) if and only if assigning p_i and p_j to f_u and g_v, respectively, satisfies (♥).

The *long code* is an error-correcting code that can be defined as the longest code over the Boolean alphabet that does not repeat bits. Precisely, it encodes a value $i \in [n]$ as the string of bits of length 2^n corresponding to the table of the function $p_i \in \mathscr{P}_2$.

Therefore, it is possible to see the problem MC as just a conjunction of Label Cover with the Long code. This conjunction has often been used before, e.g. [9, 33]. Our insight is that satisfaction of a constraint can be extended to functions that are not projections (words that are not code words of the Long code), we can simply say that the constraint corresponding to the edge (u,v) is satisfied if f_u and g_v satisfy (♥). This approach circumvents some combinatorial difficulties of using Label Cover and the Long code, and is essential for our reduction to work.

The second (and main) advantage of using identities instead of Label Cover is that it allows us to define the following promise version.

Definition 3.7. Fix a minion $\mathscr{M}$ and a positive integer N. We define $\mathrm{PMC}_{\mathscr{M}}(N)$ (*promise satisfaction of a bipartite minor condition*) as the promise problem in which, given a bipartite minor condition Σ that involves symbols of arity at most N, one needs to output YES if Σ is trivial and NO if Σ is not satisfiable in $\mathscr{M}$.

The promise in the above problem is that it is never the case that Σ is non-trivial, but satisfied in $\mathscr{M}$.

Remark 3.8. Let $\mathscr{M}_1, \mathscr{M}_2$ be two minions such that there is a minion homomorphism $\xi\colon \mathscr{M}_1 \to \mathscr{M}_2$. Then, for any N, $\mathrm{PMC}_{\mathscr{M}_2}(N)$ is obtained from $\mathrm{PMC}_{\mathscr{M}_1}(N)$ simply by strengthening the promise. To see this, observe that if some Σ is not satisfied in $\mathscr{M}_2$ then it cannot be satisfied in $\mathscr{M}_1$. Indeed, suppose the contrary, say that some f_i's and g_j's from $\mathscr{M}_1$ satisfy Σ. Since ξ is a minion homomorphism from $\mathscr{M}_1$ to $\mathscr{M}_2$, it follows that $\xi(f_i)$'s and $\xi(g_j)$'s satisfy Σ in $\mathscr{M}_2$.

We can finally formulate our second main result.

Theorem 3.9. *Let* $(\mathbf{A},\mathbf{B})$ *be a template and let* $\mathscr{M}$ *denote its polymorphism minion.*

(1) *If* N *is an upper bound on the size of any relation* $R^{\mathbf{A}}$ *of* $\mathbf{A}$ *as well as on* $|A|$*, then PCSP*$(\mathbf{A},\mathbf{B})$ *can be reduced to PMC*$_{\mathscr{M}}(N)$ *in log-space.*
(2) *For each* $N > 0$*, PMC*$_{\mathscr{M}}(N)$ *can be reduced to PCSP*$(\mathbf{A},\mathbf{B})$ *in log-space.*

Before we get to the proof, let us comment on how Theorem 3.1 follows from this result.

Proof of Theorem 3.1 given Theorem 3.9. Recall that we have two PCSP templates $(\mathbf{A}_1,\mathbf{B}_1)$ and $(\mathbf{A}_2,\mathbf{B}_2)$, and the corresponding polymorphism minions $\mathscr{M}_i = \mathrm{Pol}(\mathbf{A}_i,\mathbf{B}_i)$, $i = 1,2$. Our goal is to find a log-space reduction from $\mathrm{PCSP}(\mathbf{A}_2,\mathbf{B}_2)$ to $\mathrm{PCSP}(\mathbf{A}_1,\mathbf{B}_1)$ given that there is a minion homomorphism $\xi\colon \mathscr{M}_1 \to \mathscr{M}_2$.

By Remark 3.8, we have that, for any N, $\mathrm{PMC}_{\mathscr{M}_2}(N)$ is obtained from $\mathrm{PMC}_{\mathscr{M}_1}(N)$ by strengthening the promise. Clearly, this gives a (trivial) log-space reduction from $\mathrm{PMC}_{\mathscr{M}_2}(N)$ to $\mathrm{PMC}_{\mathscr{M}_1}(N)$.

To conclude the proof, we connect this reduction with the two reductions from Theorem 3.9. Starting with $\mathrm{PCSP}(\mathbf{A}_2,\mathbf{B}_2)$, we reduce it to $\mathrm{PMC}_{\mathscr{M}_2}(N)$ where N is given by the first item of Theorem 3.9. The above paragraph then gives us a reduction to $\mathrm{PMC}_{\mathscr{M}_1}(N)$. Finally, the second item of Theorem 3.9 provides a reduction to $\mathrm{PCSP}(\mathbf{A}_1,\mathbf{B}_1)$. □

The proof of Theorem 3.9 is given in the following two subsections.

3.2 From PCSP to Minor Conditions

We now prove Theorem 3.9(1). For that we need to provide a reduction from PCSP to PMC for a given PCSP template and its polymorphism minion. This reduction follows a standard way of proving hardness of Label Cover [1]. It is built on a two-prover protocol introduced by [10]. Our presentation of this reduction is a generalisation of [15, Lemma 4.2].

Even though we start with a PCSP with template $(\mathbf{A},\mathbf{B})$, we only use the structure $\mathbf{A}$ for the construction of a bipartite minor condition from a given instance $\mathbf{I}$ of $\mathrm{PCSP}(\mathbf{A},\mathbf{B})$. The structure $\mathbf{B}$ will influence soundness of the reduction. Fix an enumeration $A = \{a_1,\dots,a_n\}$ of the domain of $\mathbf{A}$, and consider an instance $\mathbf{I}$, i.e., a structure similar to $\mathbf{A}$. We construct a bipartite minor condition $\Sigma = \Sigma(\mathbf{A},\mathbf{I})$ over $\mathcal{U}$ and $\mathcal{V}$ in the following way.

(1) Define $\mathcal{U}$ to be the set of symbols f_v for $v \in I$, each of arity $n = |A|$.
(2) For each relation R do the following: let $k = \mathrm{ar}(R)$, $m = |R^{\mathbf{A}}|$, and let $\{(a_{\pi_1(1)},\dots,a_{\pi_k(1)}),\dots,(a_{\pi_1(m)},\dots,a_{\pi_k(m)})\}$ be the list of all tuples from $R^{\mathbf{A}}$; for each constraint $C = ((v_1,\dots,v_k),R)$, i.e., each tuple $(v_1,\dots,v_k) \in R^{\mathbf{I}}$, introduce into $\mathcal{V}$ a new symbol g_C of arity m and add to Σ the following identities:

$$f_{v_1}(x_1,\dots,x_n) \approx g_C(x_{\pi_1(1)},\dots,x_{\pi_1(m)})$$
$$\vdots$$
$$f_{v_k}(x_1,\dots,x_n) \approx g_C(x_{\pi_k(1)},\dots,x_{\pi_k(m)}).$$

This assigns, to each $\mathbf{I}$ and $\mathbf{A}$, an instance $(\Sigma,\mathcal{U},\mathcal{V})$ of $\mathrm{MC}(N)$. The bound N is the larger of $|A|$ and the maximum of $|R^{\mathbf{A}}|$, for all relations of $\mathbf{A}$. Clearly, if $\mathbf{A}$ is fixed, the condition Σ is computable from $\mathbf{I}$ in log-space.

Example 3.10. We show a reduction from NAE-Sat to MC(6). NAE-Sat is the same as $\mathrm{CSP}(\mathbf{H}_2)$ (see Example 2.8). Starting with an instance $\mathbf{I}$ of NAE-Sat, for each variable $v \in I$ we add a binary function symbol f_v, and for each constraint C involving (not necessarily different) v_1, v_2, v_3 we add a 6-ary symbol g_C and the

following identities:

$$f_{v_1}(x,y) \approx g_C(x,x,y,y,y,x),$$
$$f_{v_2}(x,y) \approx g_C(x,y,x,y,x,y),$$
$$f_{v_3}(x,y) \approx g_C(y,x,x,x,y,y).$$

The arity of f_v was chosen to be two so that the assignment of a projection would correspond to an assignment of a value to v. In particular, each variable in the above identities corresponds to one of the elements from the domain. The function g_C has arity 6, since each constraint in NAE-Sat has exactly 6 satisfying assignments; the columns of variables on the right hand side of these identities correspond to these satisfying assignments (three x's or three y's never align).

To finish the proof of Theorem 3.9(1), it is enough to prove the following lemma.

Lemma 3.11. *Let* $\mathbf{A}$, $\mathbf{B}$, *and* $\mathbf{I}$ *be similar relational structures, and let* $\Sigma = \Sigma(\mathbf{A}, \mathbf{I})$ *be constructed as above. Then*

(1) *if there is a homomorphism* $h\colon \mathbf{I} \to \mathbf{A}$, *then* Σ *is trivial; and*
(2) *if* $\mathrm{Pol}(\mathbf{A}, \mathbf{B})$ *satisfies* Σ, *then there is a homomorphism from* $\mathbf{I}$ *to* $\mathbf{B}$.

Proof. To prove item (1), assume that $h\colon \mathbf{I} \to \mathbf{A}$ is a homomorphism, and define $\zeta\colon \mathcal{U} \to \mathscr{P}_A$ by $\zeta(f_v) = p_i$ where i is chosen so that $h(v) = a_i$. We extend this assignment to symbols from $\mathcal{V}$: Let $C = ((v_1, \dots, v_k), R)$ be a constraint of $\mathbf{I}$. Since $(h(v_1), \dots, h(v_k)) \in R^{\mathbf{A}}$, we can find a unique j such that $(a_{\pi_1(j)}, \dots, a_{\pi_k(j)})$ is equal to $(h(v_1), \dots, h(v_k))$. We set $\zeta(g_C) = p_j$. Clearly, this assignment satisfies all identities of Σ involving g_C. Thus we found an assignment from $\mathcal{U} \cup \mathcal{V}$ to projections that satisfies Σ, proving that Σ is trivial.

For item (2), let us first suppose that Σ is satisfiable in projections and fix a satisfying assignment (of projections to symbols in $\mathcal{U} \cup \mathcal{V}$). In that case we can define a map from I, equivalently $\mathcal{U}$, into A by assigning to v the a_i corresponding to the projection assigned to f_v. One easy way to identify the projection is to interpret it as a projection on the set A, i.e., suppose that $\zeta\colon \mathcal{U} \to \mathscr{P}_A$ is the assignment to projections, and define

$$h(v) = \zeta(f_v)(a_1, \dots, a_n). \qquad (\clubsuit)$$

This would give a homomorphism to $\mathbf{A}$ by reversing the above argument. We only need a homomorphism to $\mathbf{B}$, but we also only have $\zeta\colon \mathcal{U} \cup \mathcal{V} \to \mathrm{Pol}(\mathbf{A}, \mathbf{B})$. Still, we define $h\colon I \to B$ by the rule ($\clubsuit$). Clearly, this is a well-defined assignment; we only need to prove that h is a homomorphism, i.e., that for each constraint $C = ((v_1, \dots, v_k), R)$, we have $(h(v_1), \dots, h(v_k)) \in R^{\mathbf{B}}$. Consider the symbol g_C and its image under ζ. We know that $\zeta(g_C)$ is a polymorphism from $\mathbf{A}$ to $\mathbf{B}$ that satisfies the corresponding identities in Σ. Let us therefore substitute a_i for x_i, for $i \in [n]$, in those identities. We obtain the following:

$$\zeta(f_{v_1})(a_1, \dots, a_n) = \zeta(g_C)(a_{\pi_1(1)}, \dots, a_{\pi_1(m)})$$
$$\vdots$$
$$\zeta(f_{v_k})(a_1, \dots, a_n) = \zeta(g_C)(a_{\pi_k(1)}, \dots, a_{\pi_k(m)}).$$

But since all the tuples $(a_{\pi_1(j)}, \dots, a_{\pi_k(j)})$, for $j \in [m]$, are in $R^{\mathbf{A}}$ and $\zeta(g_C)$ is a polymorphism from $\mathbf{A}$ to $\mathbf{B}$, we get that the resulting tuple $(h(v_1), \dots, h(v_k))$ is in $R^{\mathbf{B}}$. □

Remark 3.12. Note that if we take $\mathbf{A} = \mathbf{B}$ in the previous lemma, we obtain that $\Sigma(\mathbf{A}, \mathbf{I})$ is trivial if and only if $\Sigma(\mathbf{A}, \mathbf{I})$ is satisfied in $\mathrm{Pol}(\mathbf{A})$ if and only if $\mathbf{I}$ maps homomorphically to $\mathbf{A}$.

3.3 From Minor Conditions to PCSP

Our proof of Theorem 3.9(2) follows another standard reduction in approximation: a conjunction of Long code testing and Label Cover [9]. If both $\mathbf{A}$ and $\mathbf{B}$ are Boolean ($A = B = \{0, 1\}$), the construction can be viewed as a certain Long code test. Such analogy fails when A and B have different sizes. Nevertheless, projections on the set A (as opposed to $\{0, 1\}$ for long codes) still play a crucial role in the completeness of this reduction. The reduction has also appeared many times in the presented algebraic form and seems to be folklore (see e.g. [23, Lemma 3.8]).

The key idea of the construction is that the question 'Is this bipartite minor condition satisfied by polymorphisms of $\mathbf{A}$?' can be interpreted as an instance of CSP($\mathbf{A}$). The main ingredient is that a polymorphism is a homomorphism from $\mathbf{A}^n$; this gives an instance whose solutions are exactly n-ary polymorphisms. Such instances are sometimes called "indicator instances" [37]. By considering the union of several such instances (one for each function symbol appearing in Σ) and then introducing equality constraints that reflect the identities, we get that a solution to the obtained instance corresponds to polymorphisms satisfying the identities. In detail, let us fix a template $(\mathbf{A}, \mathbf{B})$ and a bound on arity N. We start with a bipartite minor condition $(\Sigma, \mathcal{U}, \mathcal{V})$ with arity bounded by N, and construct an instance $\mathbf{I} = \mathbf{I}_\Sigma(\mathbf{A})$ of PCSP($\mathbf{A}, \mathbf{B}$) in three steps:

(1) for each n-ary symbol f in $\mathcal{U} \cup \mathcal{V}$, introduce into $\mathbf{I}$ a fresh copy of $\mathbf{A}^n$ where each element $(a_1, \dots, a_n) \in A^n$ is replaced by a new element denoted by $v_{f(a_1,\dots,a_n)}$;
(2) for each identity $f(x_1, \dots, x_l) \approx g(x_{\pi(1)}, \dots, x_{\pi(r)})$ in Σ, and $a_1, \dots, a_l \in A$, add an equality constraint ensuring $v_{f(a_1,\dots,a_l)} = v_{g(a_{\pi(1)},\dots,a_{\pi(r)})}$;
(3) identify all pairs of variables connected by (a path of) equality constraints and then remove the equality constraints.

Clearly, the first and the second step can be done in log-space (note that the arity n is bounded by the constant N). The third step can be done in log-space by [45].

Lemma 3.13. *Let* $(\mathbf{A}, \mathbf{B})$ *be a template,* $N > 0$, *and let* $\mathscr{M} = \mathrm{Pol}(\mathbf{A}, \mathbf{B})$. *The above construction gives a log-space reduction from* $PMC_{\mathscr{M}}(N)$ *to PCSP*$(\mathbf{A}, \mathbf{B})$.

Proof. To prove completeness, suppose that Σ is trivial, i.e., there is a mapping $\zeta\colon \mathcal{U} \cup \mathcal{V} \to \mathscr{P}_A$ which satisfies Σ. We define a homomorphism from $\mathbf{I}$ to $\mathbf{A}$ by setting

$$h(v_{f(a_1,\dots,a_n)}) = \zeta(f)(a_1, \dots, a_n).$$

Note that h is well-defined since ζ satisfies Σ. Further, since every projection is a polymorphism of $\mathbf{A}$, h is a homomorphism.

To prove soundness, assume that there is a homomorphism $h\colon \mathbf{I} \to \mathbf{B}$. We reverse the above argument and define ζ by the rule $\zeta(f)(a_1, \dots, a_n) = h(v_{f(a_1,\dots,a_n)})$ for all $a_1, \dots, a_n \in A$. Now, $\zeta(f) \in \mathscr{M}$ follows from the first step of the construction of $\mathbf{I}$, and satisfaction of the identities from Σ follows from the second and the third steps of the construction. □

This concludes the proof of Theorem 3.9.

4 NP-HARDNESS OF $(2k-1)$-COLOURING k-COLOURABLE GRAPHS

In this section we prove the following theorem, called result (C) in the introduction.

Theorem 4.1. *For any $k \geq 3$, it is NP-hard to distinguish k-colourable graphs from those that are not $(2k-1)$-colourable.*

For better accessibility, we give a direct proof which mimics the proofs from the previous section for this special case. At the end of this section we mention a more general result (presented in the full version [22]) whose proof uses more of our general theory.

As a starting point for our reduction, we will use the following result on the hardness of approximate hypergraph colouring.

Theorem 4.2 ([28]). *For any $c \geq 2$, it is NP-hard to distinguish between 3-uniform hypergraphs that are 2-colourable and those that are not even c-colourable.*

As we noted in Example 2.10, this theorem translates directly into NP-hardness of $\mathrm{PCSP}(\mathbf{H}_2, \mathbf{H}_c)$ where $\mathbf{H}_c = (E_c; \mathrm{NAE}_c)$.

Proof. Fix $k \geq 3$ and let c be the number of binary polymorphisms from $\mathbf{K}_k$ to $\mathbf{K}_{2k-1}$ (i.e., of $(2k-1)$-colourings of $\mathbf{K}_k^2$). Let $\mathbf{H} = (V_H, E_H)$ be a 3-uniform hypergraph. We will show how to construct, in polynomial time, a graph $\mathbf{I}$ such that if $\mathbf{H}$ is 2-colourable then $\mathbf{I}$ is k-colourable and if $\mathbf{I}$ is $(2k-1)$-colourable then $\mathbf{H}$ is c-colourable. This will prove the theorem.

Consider the following system Σ of identities (essentially the same as introduced in Example 3.10). For each $v \in V_H$, we introduce a binary function symbol f_v, and for each $e = (v_1, v_2, v_3) \in E_H$, we introduce a 6-ary symbol g_e. Let Σ_e be the following system of identities:

$$f_{v_1}(x,y) \approx g_e(x,x,y,y,y,x),$$
$$f_{v_2}(x,y) \approx g_e(x,y,x,y,x,y),$$
$$f_{v_3}(x,y) \approx g_e(y,x,x,x,y,y).$$

Note that the vertices in a hyperedge can be ordered arbitrarily to produce the above identities. Finally, let $\Sigma = \bigcup_e \Sigma_e$. Note that $\Sigma = \Sigma(\mathbf{H}_2, \mathbf{H})$, following the construction in Section 3.2.

We now turn Σ into a graph $\mathbf{I} = \mathbf{I}_\Sigma(\mathbf{K}_k)$, following the 3-step construction in Section 3.3. Specifically, let $\mathbf{I} = (V_I, E_I)$ where V_I initially consists of all vertices of the form $u_{f_v(a_1,a_2)}$ and of the form $u_{g_e(a_1,\ldots,a_6)}$ where $v \in V_H$, $e \in E_H$, and $a_i \in E_k$ for all i.

For each $v \in V_H$, and each edge $((a_1,a_2),(b_1,b_2))$ of $\mathbf{K}_k^2$ (i.e., $a_i \neq b_i$ for $i = 1,2$), add an edge between $u_{f_v(a_1,a_2)}$ and $u_{f_v(b_1,b_2)}$ to E_I. Similarly, for each $e \in V_H$, and each edge $((a_1,\ldots,a_6),(a_1,\ldots,a_6))$ of $\mathbf{K}_k^6$, add an edge between $u_{g_e(a_1,\ldots,a_6)}$ and $u_{g_e(b_1,\ldots,b_6)}$ to E_G.

Finally, we identify (glue together) a number of vertices. Namely, for each identity in Σ of the form $f_v(x,y) \approx g_e(x,x,y,y,y,x)$, and each $a, b \in E_k$, identify vertices $u_{f_v(a,b)}$ and $u_{g_e(a,a,b,b,b,a)}$, and perform a similar procedure for the other two types of identities in Σ. This finishes the construction of $\mathbf{I}$, and it is clear that it can be done in polynomial time in the size of $\mathbf{H}$. In the rest of the proof, we show that $\mathbf{I}$ has the two required properties.

Assume first that $\mathbf{H}$ is 2-colourable. Call the 2 colours x and y, and let $\beta\colon V_H \to \{x,y\}$ be a 2-colouring. Define $i_v = 1$ if $\beta(v) = x$ and $i_v = 2$ if $\beta(v) = y$. Next, for every $e = (v_1,v_2,v_3) \in E_H$, let $i_e \in [1,6]$ be the number of the column on the right-hand side of Σ_e such that $(\beta(v_1),\beta(v_2),\beta(v_3))$ is that column. It is easy to see that this number is well-defined. We then define the following mapping β' from I to K_k. For each $v \in V_H$ and each $(a_1,a_2) \in E_k^2$, let $\beta'(u_{f_v(a_1,a_2)}) = a_{i_v}$. Similarly, for each $e \in E_H$ and each $(a_1,\ldots,a_6) \in E_k^6$, let $\beta'(u_{g_e(a_1,\ldots,a_6)}) = a_{i_e}$. It is straightforward to check that β' is well-defined (i.e., if two vertices were glued in the construction of $\mathbf{I}$ then they get the same colour) and it is a proper k-colouring of $\mathbf{I}$.

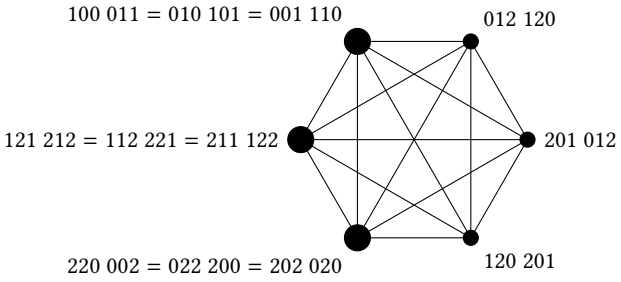

Figure 1: A 6-clique in the graph G for $k = 3$. The vertices on the left were involved in the gluing that produced G.

Now let us assume that $\mathbf{I}$ is $(2k-1)$-colourable and $\gamma\colon V_I \to E_{2k-1}$ is such a colouring. One important (and well-known) observation about the construction of $\mathbf{I}$ is that, for any $t \geq k$ and any fixed vertex $v \in V_H$, any t-colouring α of the subgraph of $\mathbf{I}$ induced by all vertices of the form $u_{f_v(a,b)}$ gives rise to a binary polymorphism f from $\mathbf{K}_k$ to $\mathbf{K}_t$ defined by $f(a,b) = \alpha(u_{f_v(a,b)})$. Similarly, any t-colouring α of the subgraph of $\mathbf{I}$ induced by all vertices of the form $u_{g_e(\mathbf{a})}$, for a fixed e, gives rise to a 6-ary polymorphism from $\mathbf{K}_k$ to $\mathbf{K}_t$. These facts easily follow directly from definitions.

By the observation above, the colouring γ gives rise to binary polymorphisms $f_v, v \in V_H$, and 6-ary polymorphisms $g_e, e \in E_H$, from $\mathbf{K}_k$ to $\mathbf{K}_{2k-1}$. Moreover, these polymorphisms satisfy all identities in Σ due to the gluings made when constructing $\mathbf{I}$. Now consider the c possible binary polymorphisms from $\mathbf{K}_k$ to $\mathbf{K}_{2k-1}$ as colours and define a c-colouring γ' of $\mathbf{H}$ by $\gamma'(v) = f_v$. It remains to prove that γ' is a proper colouring of $\mathbf{H}$. Assume, for contradiction, that it is not, so there is a hyperedge $e = (v_1,v_2,v_3) \in E_H$ such that $f_{v_1} = f_{v_2} = f_{v_3}$. This means that the 6-ary polymorphism $g = g_e \in \mathrm{Pol}(\mathbf{K}_k, \mathbf{K}_{2k-1})$ satisfies the following identities: $g(x,x,y,y,y,x) \approx g(x,y,x,y,x,y) \approx g(y,x,x,x,y,y)$. To finish the proof, we now show that such a g does not exist.

The polymorphism g can viewed as $(2k-1)$-colouring of $\mathbf{K}_k^6$. By assumption, for each $(a,b) \in K_k^2$, this colouring gives the same colour (which may depend on (a,b)) to every triple of vertices of the form (a,a,b,b,b,a), (a,b,a,b,a,b), and (b,a,a,a,b,b). Thus, g can be viewed as a $(2k-1)$-colouring of the graph $\mathbf{G}$ obtained from $\mathbf{K}_k^6$ by gluing together, for each (a,b), the three vertices of the above form. We claim that then $\mathbf{G}$ contains a $(2k)$-clique, which would contradict the assumption that $\mathbf{G}$ is $(2k-1)$-colourable. For all $i \in K_k$, we consider:

$$\mathbf{a}_i = (i, i+1, i+2, i+1, i+2, i) \text{ and } \mathbf{b}_i = (i+1, i, i, i, i+1, i+1)$$

where addition is modulo k. Note that, when producing the graph $\mathbf{G}$, each vertex $\mathbf{b}_i$ was identified with both $(i, i+1, i, i+1, i, i+1)$ and $(i, i, i+1, i+1, i+1, i)$. We claim that the set $\{\mathbf{a}_i, \mathbf{b}_i \mid i \in K_k\}$ induces a $(2k)$-clique in $\mathbf{G}$. Fig. 1 depicts this clique in $\mathbf{G}$ for $k = 3$. Clearly for any $i \neq j$, there is an edge between $\mathbf{a}_i$ and $\mathbf{a}_j$, as well as between $\mathbf{b}_i$ and $\mathbf{b}_j$ (in $\mathbf{K}_k^6$, and hence in $\mathbf{G}$).

For edges between $\mathbf{a}_i$ and $\mathbf{b}_j$, we consider the following cases:

- $j \notin \{i, i+1\}$. Then also $j+1 \notin \{i+1, i+2\}$, so there is an edge between $\mathbf{b}_j$ and $\mathbf{a}_i$ since $\mathbf{b}_j$ was identified with $(j, j, j+1, j+1, j+1, j)$.
- $j = i$. There is an edge between $\mathbf{a}_i$ and $\mathbf{b}_j$ since $\mathbf{b}_j$ was identified with $(j+1, j, j, j, j+1, j+1)$.
- $j = i+1$. There is an edge between $\mathbf{a}_i$ and $\mathbf{b}_j$ since $\mathbf{b}_j$ was identified with $(j, j+1, j, j+1, j, j+1)$.

This concludes the proof. □

A 6-ary function g satisfying the identities $g(x,x,y,y,y,x) \approx g(x,y,x,y,x,y) \approx g(y,x,x,x,y,y)$ is called an Olšák function. The key in the above proof is that $\mathrm{Pol}(\mathbf{K}_k, \mathbf{K}_{2k-1})$ does not contain an Olšák function. In [22], we show that, for any PCSP template $(\mathbf{A}, \mathbf{B})$, the absence of an Olšák function in $\mathrm{Pol}(\mathbf{A}, \mathbf{B})$ is equivalent to the existence of a minion homomorphism from $\mathrm{Pol}(\mathbf{A}, \mathbf{B})$ to $\mathrm{Pol}(\mathbf{H}_2, \mathbf{H}_c)$ for some $c \geq 2$. By Theorem 3.1 and Theorem 4.2, this implies that any PCSP without an Olšák polymorphism is NP-hard. Finally, we remark that, when $c \geq 2k$, $\mathrm{Pol}(\mathbf{K}_k, \mathbf{K}_c)$ does have an Olšák polymorphism. This and further properties of $\mathrm{Pol}(\mathbf{K}_k, \mathbf{K}_c)$ can be found in [22].

5 CONCLUSION

This paper deals with the Promise CSP framework, which is a significant generalisation of the (finite-domain) CSP. The PCSP provides a nice interplay between the study of approximability and universal-algebraic methods in computational complexity. We presented a general abstract algebraic theory that captures the complexity of PCSPs with a fixed template $(\mathbf{A}, \mathbf{B})$. The key element in our approach is the bipartite minor conditions satisfied in the polymorphism minion $\mathrm{Pol}(\mathbf{A}, \mathbf{B})$. We have shown that such conditions determine the complexity $\mathrm{PCSP}(\mathbf{A}, \mathbf{B})$. We gave some applications of our general theory, in particular, in approximate graph colouring.

The complexity landscape of PCSP (beyond CSP) is largely unknown, even in the Boolean case (despite some progress in [16]), and includes many specific problems of interest. We hope that our theory will provide the basis for a fruitful research programme of charting this landscape. Below we discuss some of the possible directions within this programme.

Let us first discuss how the complexity classification quest for PCSPs compares with that for CSPs. As we said above, the gist of the algebraic approach is that lack or presence of (high-dimensional) symmetries determines the complexity. For CSPs, there is a sharp algebraic dichotomy: having only trivial symmetries (i.e., satisfying only those systems of minor identities that are satisfied in polymorphisms of every CSP) leads to NP-hardness, while any non-trivial symmetry implies rather strong symmetry and thus leads to tractability. Moreover, the algorithms for tractable cases are (rather involved) combinations of only two basic algorithms — one is based on local propagation [5] and the other can be seen as a very general form of Gaussian elimination [36]. It is already clear that the situation is more complicated for PCSPs: there are hard PCSPs with non-trivial (but limited in some sense) symmetries, and tractable cases are more varied [3, 15, 16, 28]. This calls for more advanced methods, and we hope that our paper will provide the basis for such methods. There is an obvious question whether PCSPs exhibit a dichotomy as CSPs do, but there is not enough evidence yet to conjecture an answer. More specifically, it is not clear whether there is any PCSP whose polymorphisms are not limited enough (in terms of satisfying systems of minor identities) to give NP-hardness, but also not strong enough to ensure tractability. Classifications for special cases such as Boolean PCSPs and graph homomorphisms would help to obtain more intuition about the general complexity landscape of PCSPs, but these special cases are currently open.

The sources of hardness in PCSP appear to be much more varied than in CSP (that has a unique such source), and much remains to be understood there. What limitations on the bipartite minor conditions satisfied in polymorphisms lead to NP-hardness? Some general and some specific results in this direction can be found in the full version [22], but it is clear that our general theory needs to be further developed. Currently, variants of the Gap Label Cover provide the source of hardness, but it is possible that new versions of GLC may need to be used. It is not clear in advance what these versions would be — their form would be dictated by the analysis of polymorphisms and minor conditions. However, it would interesting to eventually go even further and avoid dependency on deep approximation results such as the PCP theorem and parallel repetition. Instead, one would aim to provide a self-contained theory that includes a new type of reduction between PCSPs so that one could bypass the PCP theorem and parallel repetition altogether. In particular, can one come up with purely algebraic reductions (e.g. by extending pp-constructions) that create and amplify the algebraic gap in problems $\mathrm{PMC}_{\mathscr{M}}(N)$, which is what the PCP theorem and parallel repetition do for the quantitative gap in Gap Label Cover?

The analysis of polymorphisms of approximate graph colouring problems (and their relatives) may provide further intuition as to what limitations on minor conditions can be used for hardness proofs.

What algorithmic techniques are needed to solve tractable PCSPs? One general approach to proving tractability of $\mathrm{PCSP}(\mathbf{A}, \mathbf{B})$ is presented in [16]. The main idea is to find a structure $\mathbf{D}$ such that $\mathbf{A} \to \mathbf{D} \to \mathbf{B}$ and $\mathrm{CSP}(\mathbf{D})$ is tractable. Such structures are called "homomorphic sandwiches" in [16]. It is clear that an algorithm for $\mathrm{CSP}(\mathbf{D})$ solves $\mathrm{PCSP}(\mathbf{A}, \mathbf{B})$. In general, $\mathbf{D}$ may have infinite domain (but instances of $\mathrm{CSP}(\mathbf{D})$ are still finite) — indeed, for $\mathrm{PCSP}(\mathbf{T}, \mathbf{H}_2)$ from Example 2.8 no such finite $\mathbf{D}$ exists [4], but one infinite example is $\mathbf{D} = (\mathbb{Z}, x + y + z = 1)$ [15, 16]. A range of examples when this works, particularly with $\mathbf{D}$ relating to linear programming and affine relaxations, is given in [16]. It would be very interesting to develop a general theory of how such a structure $\mathbf{D}$ can be constructed from bipartite minor conditions satisfied in $\mathrm{Pol}(\mathbf{A}, \mathbf{B})$ and what properties of such conditions guarantee tractability of $\mathrm{CSP}(\mathbf{D})$. In general, infinite-domain CSPs have a considerably more complicated structure than their finite counterparts; in particular, they exhibit no dichotomy [12]. We remark that the algebraic theory of infinite-domain CSPs is being developed [11, 44] and it often has an

additional model-theoretic flavour. It is another interesting feature of the (finite-domain) PCSP framework that it seems to require research into infinite-domain CSP.

Acknowledgements

We would like to thank Libor Barto, Mirek Olšák, Alex Kazda, Josh Brakensiek, Venkat Guruswami, Erkko Lehtonen, and Marcello Mamino for valuable discussions. We also thank anonymous referees for useful suggestions.

REFERENCES

[1] Sanjeev Arora, László Babai, Jacques Stern, and Z. Sweedyk. 1997. The Hardness of Approximate Optima in Lattices, Codes, and Systems of Linear Equations. *J. Comput. Syst. Sci.* 54, 2 (April 1997), 317–331.

[2] Per Austrin, Amey Bhangale, and Aditya Potukuchi. 2018. Improved Inapproximability of Rainbow Coloring. *ArXiv e-prints* (Oct. 2018). arXiv:1810.02784

[3] Per Austrin, Venkatesan Guruswami, and Johan Håstad. 2017. $(2+\varepsilon)$-Sat Is NP-hard. *SIAM J. Comput.* 46, 5 (2017), 1554–1573.

[4] Libor Barto. 2019. Promises make finite (constraint satisfaction) problems infinitary. (2019). to appear in LICS 2019.

[5] Libor Barto and Marcin Kozik. 2014. Constraint Satisfaction Problems Solvable by Local Consistency Methods. *J. ACM* 61, 1 (Jan. 2014), 3:1–3:19.

[6] Libor Barto and Marcin Kozik. 2016. Robustly Solvable Constraint Satisfaction Problems. *SIAM J. Comput.* 45, 4 (2016), 1646–1669.

[7] Libor Barto, Andrei Krokhin, and Ross Willard. 2017. Polymorphisms, and How to Use Them. In *The Constraint Satisfaction Problem: Complexity and Approximability*, Andrei Krokhin and Stanislav Živný (Eds.). Dagstuhl Follow-Ups, Vol. 7. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany, 1–44.

[8] Libor Barto, Jakub Opršal, and Michael Pinsker. 2018. The wonderland of reflections. *Israel Journal of Mathematics* 223, 1 (Feb 2018), 363–398.

[9] Mihir Bellare, Oded Goldreich, and Madhu Sudan. 1998. Free Bits, PCPs, and Nonapproximability—Towards Tight Results. *SIAM J. Comput.* 27, 3 (1998), 804–915.

[10] Michael Ben-Or, Shafi Goldwasser, Joe Kilian, and Avi Wigderson. 1988. Multi-prover Interactive Proofs: How to Remove Intractability Assumptions. In *Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing (STOC '88)*. ACM, New York, NY, USA, 113–131.

[11] Manuel Bodirsky. 2008. Constraint Satisfaction Problems with Infinite Templates. In *Complexity of Constraints* (2009-05-05) *(Lecture Notes in Computer Science)*, Nadia Creignou, Phokion G. Kolaitis, and Heribert Vollmer (Eds.), Vol. 5250. Springer, 196–228.

[12] Manuel Bodirsky and Martin Grohe. 2008. Non-dichotomies in Constraint Satisfaction Complexity. In *Automata, Languages and Programming, 35th International Colloquium, ICALP 2008, Reykjavik, Iceland, July 7-11, 2008, Proceedings, Part II*. 184–196.

[13] Joshua Brakensiek and Venkatesan Guruswami. 2016. New Hardness Results for Graph and Hypergraph Colorings. In *31st Conference on Computational Complexity (CCC 2016) (Leibniz International Proceedings in Informatics (LIPIcs))*, Ran Raz (Ed.), Vol. 50. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany, 14:1–14:27.

[14] Joshua Brakensiek and Venkatesan Guruswami. 2017. The Quest for Strong Inapproximability Results with Perfect Completeness. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2017, August 16-18, 2017, Berkeley, CA, USA*. 4:1–4:20.

[15] Joshua Brakensiek and Venkatesan Guruswami. 2018. Promise Constraint Satisfaction: Structure Theory and a Symmetric Boolean Dichotomy. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'18)*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1782–1801.

[16] Joshua Brakensiek and Venkatesan Guruswami. 2019. An Algorithmic Blend of LPs and Ring Equations for Promise CSPs. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, San Diego, California, USA, January 6-9, 2019*. 436–455.

[17] Jonah Brown-Cohen and Prasad Raghavendra. 2015. Combinatorial Optimization Algorithms via Polymorphisms. *CoRR* abs/1501.01598 (2015).

[18] Andrei Bulatov and Peter Jeavons. 2001. *Algebraic structures in combinatorial problems.* Technical Report.

[19] Andrei Bulatov, Peter Jeavons, and Andrei Krokhin. 2005. Classifying the Complexity of Constraints Using Finite Algebras. *SIAM J. Comput.* 34, 3 (March 2005), 720–742.

[20] Andrei A. Bulatov. 2013. The Complexity of the Counting Constraint Satisfaction Problem. *J. ACM* 60, 5, Article 34 (Oct. 2013), 41 pages.

[21] Andrei A. Bulatov. 2017. A Dichotomy Theorem for Nonuniform CSPs. In *2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)*. 319–330.

[22] Jakub Bulín, Andrei Krokhin, and Jakub Opršal. 2018. Algebraic approach to promise constraint satisfaction. *ArXiv e-prints* (2018). arXiv:1811.00970

[23] Hubie Chen and Benoit Larose. 2017. Asking the Metaquestions in Constraint Tractability. *ACM Trans. Comput. Theory* 9, 3 (Oct. 2017), 11:1–11:27.

[24] Hubie Chen, Matthew Valeriote, and Yuichi Yoshida. 2016. Testing Assignments to Constraint Satisfaction Problems. In *IEEE 57th Annual Symposium on Foundations of Computer Science, FOCS 2016, 9-11 October 2016, Hyatt Regency, New Brunswick, New Jersey, USA*. 525–534.

[25] Víctor Dalmau, Marcin Kozik, Andrei A. Krokhin, Konstantin Makarychev, Yury Makarychev, and Jakub Opršal. 2017. Robust algorithms with polynomial loss for near-unanimity CSPs. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017, Barcelona, Spain, Hotel Porta Fira, January 16-19*. 340–357.

[26] Victor Dalmau, Andrei Krokhin, and Rajsekar Manokaran. 2018. Towards a characterization of constant-factor approximable finite-valued CSPs. *J. Comput. System Sci.* 97, 14–27.

[27] Irit Dinur, Elchanan Mossel, and Oded Regev. 2009. Conditional Hardness for Approximate Coloring. *SIAM J. Comput.* 39, 3 (2009), 843–873.

[28] Irit Dinur, Oded Regev, and Clifford Smyth. 2005. The Hardness of 3-Uniform Hypergraph Coloring. *Combinatorica* 25, 5 (Sep 2005), 519–535.

[29] Tomás Feder and Moshe Y. Vardi. 1998. The Computational Structure of Monotone Monadic SNP and Constraint Satisfaction: A Study Through Datalog and Group Theory. *SIAM J. Comput.* 28, 1 (Feb. 1998), 57–104.

[30] M. R. Garey and David S. Johnson. 1976. The Complexity of Near-Optimal Graph Coloring. *J. ACM* 23, 1 (1976), 43–49.

[31] Venkatesan Guruswami and Sanjeev Khanna. 2004. On the Hardness of 4-Coloring a 3-Colorable Graph. *SIAM Journal on Discrete Mathematics* 18, 1 (2004), 30–40.

[32] Venkatesan Guruswami and Euiwoong Lee. 2017. Strong inapproximability results on balanced rainbow-colorable hypergraphs. *Combinatorica* (14 Dec 2017).

[33] Johan Håstad. 2001. Some Optimal Inapproximability Results. *J. ACM* 48, 4 (July 2001), 798–859.

[34] Pavol Hell and Jaroslav Nešetřil. 1990. On the complexity of H-coloring. *J. Combin. Theory Ser. B* 48, 1 (1990), 92–110.

[35] Sangxia Huang. 2013. Improved Hardness of Approximating Chromatic Number. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques: 16th International Workshop, APPROX 2013, and 17th International Workshop, RANDOM 2013, Berkeley, CA, USA, August 21-23, 2013. Proceedings*, Prasad Raghavendra, Sofya Raskhodnikova, Klaus Jansen, and José D. P. Rolim (Eds.). Springer, Berlin, Heidelberg, 233–243.

[36] Paweł M. Idziak, Petar Marković, Ralph McKenzie, Matthew Valeriote, and Ross Willard. 2010. Tractability and Learnability Arising from Algebras with Few Subpowers. *SIAM J. Comput.* 39, 7 (2010), 3023–3037.

[37] Peter Jeavons, David Cohen, and Marc Gyssens. 1997. Closure Properties of Constraints. *J. ACM* 44, 4 (July 1997), 527–548.

[38] Ken-ichi Kawarabayashi and Mikkel Thorup. 2017. Coloring 3-Colorable Graphs with Less than $n^{1/5}$ Colors. *J. ACM* 64, 1 (2017), 4:1–4:23.

[39] Sanjeev Khanna, Nathan Linial, and Shmuel Safra. 2000. On the Hardness of Approximating the Chromatic Number. *Combinatorica* 20, 3 (01 Mar 2000), 393–415.

[40] Subhash Khot. 2001. Improved inapproximability results for MaxClique, chromatic number and approximate graph coloring. In *Foundations of Computer Science, 2001. Proceedings. 42nd IEEE Symposium on*. IEEE, 600–609.

[41] Vladimir Kolmogorov, Andrei Krokhin, and Michal Rolínek. 2017. The complexity of general-valued CSPs. *SIAM J. Comput.* 46, 3 (2017), 1087–1110.

[42] Andrei Krokhin and Stanislav Živný (Eds.). 2017. *The Constraint Satisfaction Problem: Complexity and Approximability*. Dagstuhl Follow-Ups, Vol. 7. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.

[43] Colin McDiarmid. 1993. A Random Recolouring Method for Graphs and Hypergrams. *Combinatorics, Probability & Computing* 2 (1993), 363–365.

[44] Michael Pinsker. 2015. Algebraic and model theoretic methods in constraint satisfaction. *ArXiv e-prints* (2015). arXiv:1507.00931

[45] Omer Reingold. 2008. Undirected Connectivity in Log-space. *J. ACM* 55, 4 (Sept. 2008), 17:1–17:24.

[46] Thomas J. Schaefer. 1978. The Complexity of Satisfiability Problems. In *Proceedings of the Tenth Annual ACM Symposium on Theory of Computing (STOC '78)*. ACM, New York, NY, USA, 216–226.

[47] Johan Thapper and Stanislav Živný. 2016. The Complexity of Finite-Valued CSPs. *J. ACM* 63, 4 (2016), 37:1–37:33.

[48] Dmitriy Zhuk. 2017. A Proof of CSP Dichotomy Conjecture. In *2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)*. 331–342.